CodeBlocks调试功能快捷教程

制作　贺利坚

贺利坚　Email: sxhelijian@163.com

为什么要用单步调试

- 单步调试是发现运行错误和逻辑错误的“利器”，可用于

（1）跟踪程序的执行流程，发现错误的线索

——发现该走A路径，却走了B路径

（2）跟踪过程中，还可以观察变量的变化，从而发现其中存在的问题

——该是 1，执行中却分明是 2，之前哪儿出了问题？

- 单步执行除了可以帮助我们发现错误，对于初学者，还可以帮助我们理解语言的机制。
- “工欲善其事，必先利其器”，单步调试就是程序设计者最重要的工具之一
  - 这种**工具**的形态是软件。**程序员用软件当工具**，正常得不得了。用好这种工具！

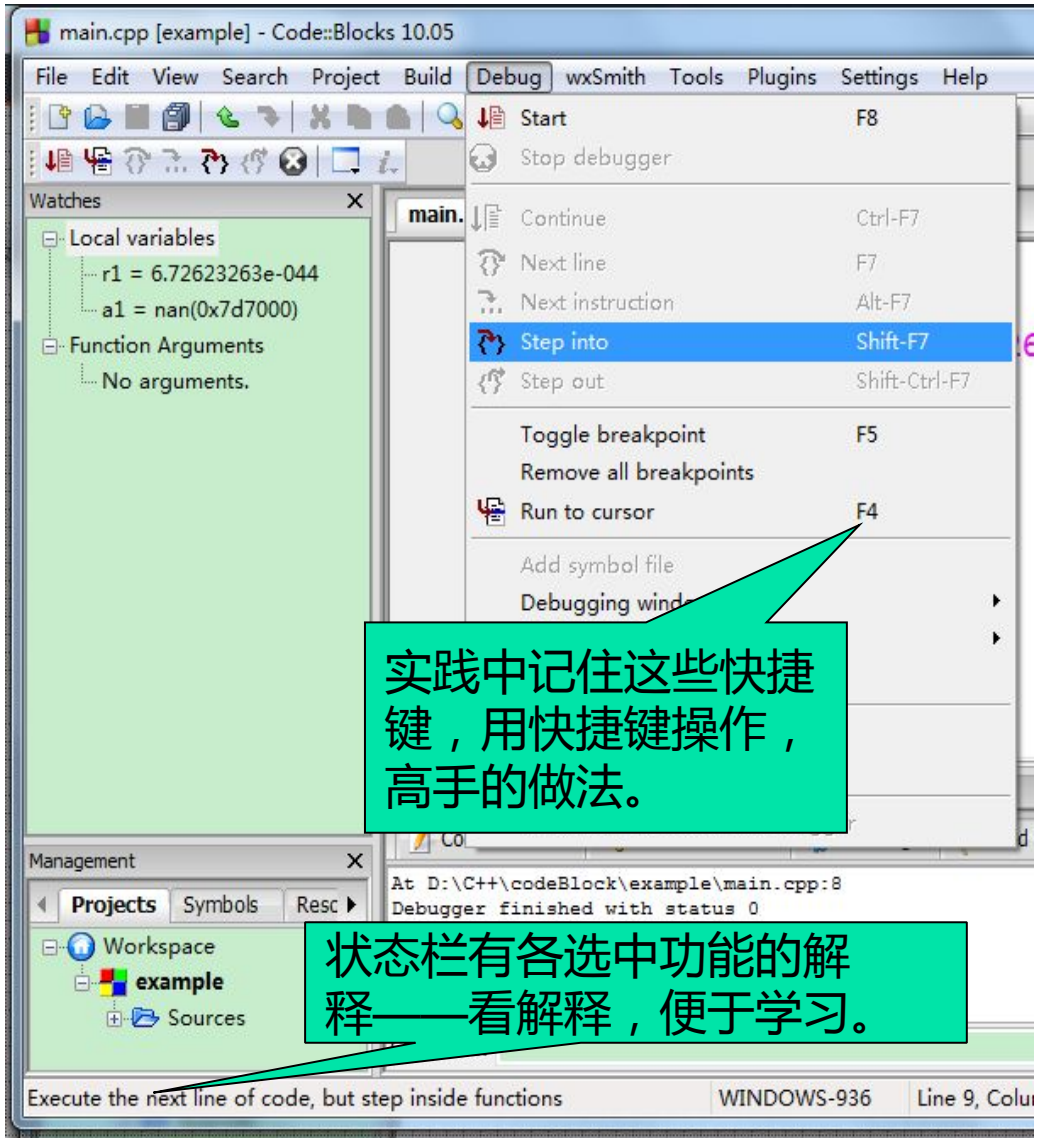
寄语初学者

- 对于初学者，掌握所用的集成开发环境的一般用法，是一件非常重要的事情。
- 单步调试更是在实践中掌握的，做中学，学中记得用
- 本文讲Code::Blocks的单步调试，其他开发环境，也有类似功能，通一百通。

- 打开你的Code::Blocks，一边看，一边练……

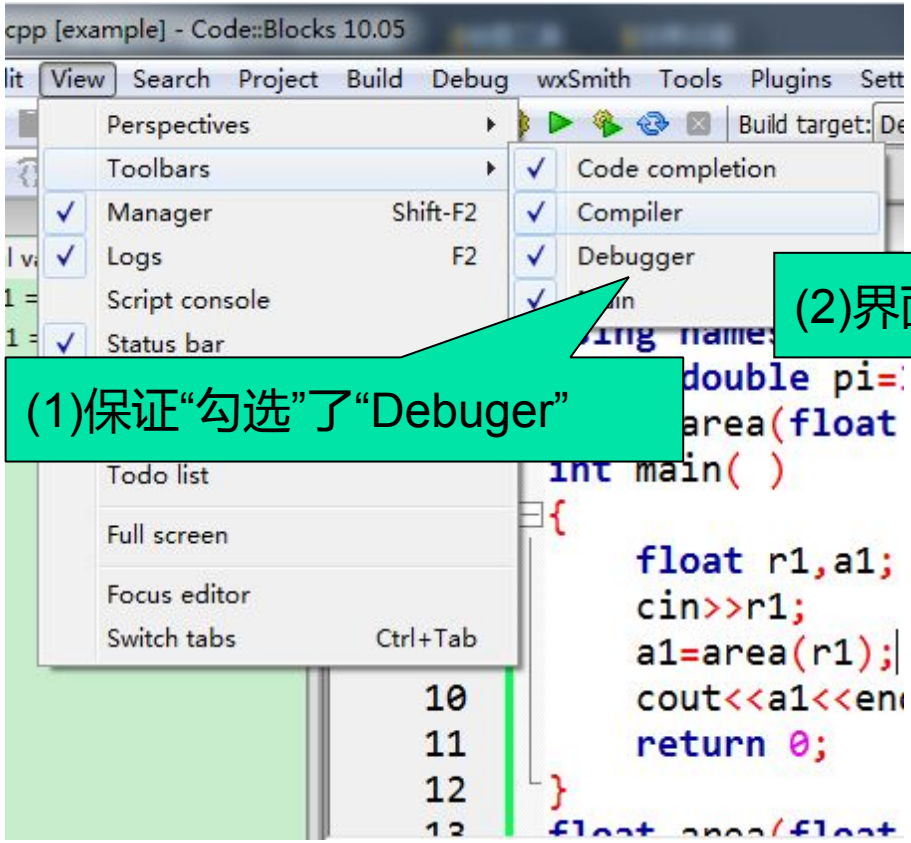
先认识菜单

- 单步调试功能在Debug菜单中
- 日常应用常用工具栏或快捷键，举步艰难时想起菜单，这里有全部的功能！
- 观察菜单，知道有哪些功能；
- 灰色的部分，会在“条件”具备时变得可用。



单步调试的工具栏

- 用工具栏中按钮，而不是菜单，更便捷
- 如何“找”出工具栏？



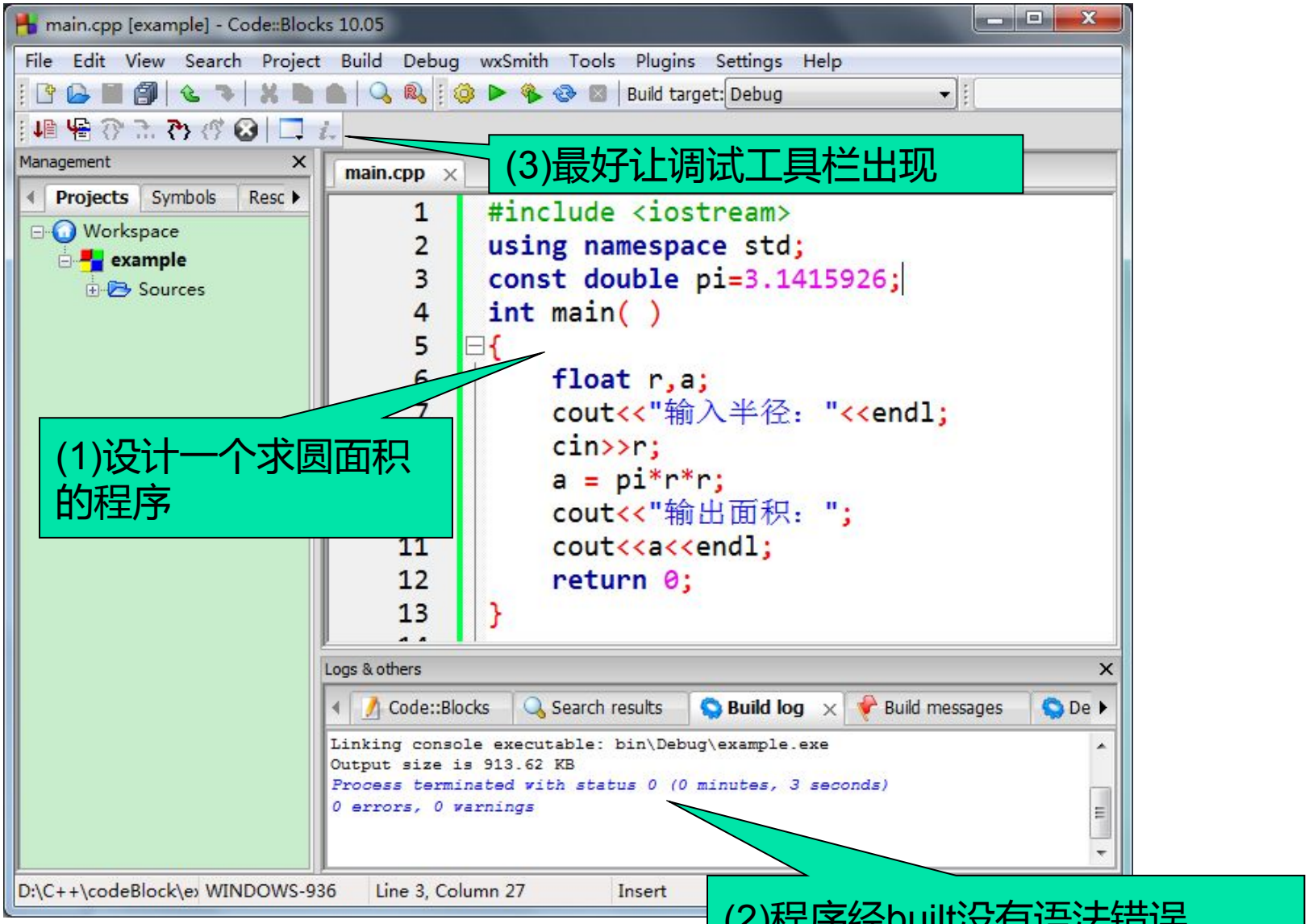
- 将鼠标“浮”在工具栏按钮上，会看到Run to Cursor、Next Line——菜单也有
- 但，这儿更便捷！

技能1：用Run to Cursor、Next Line调试

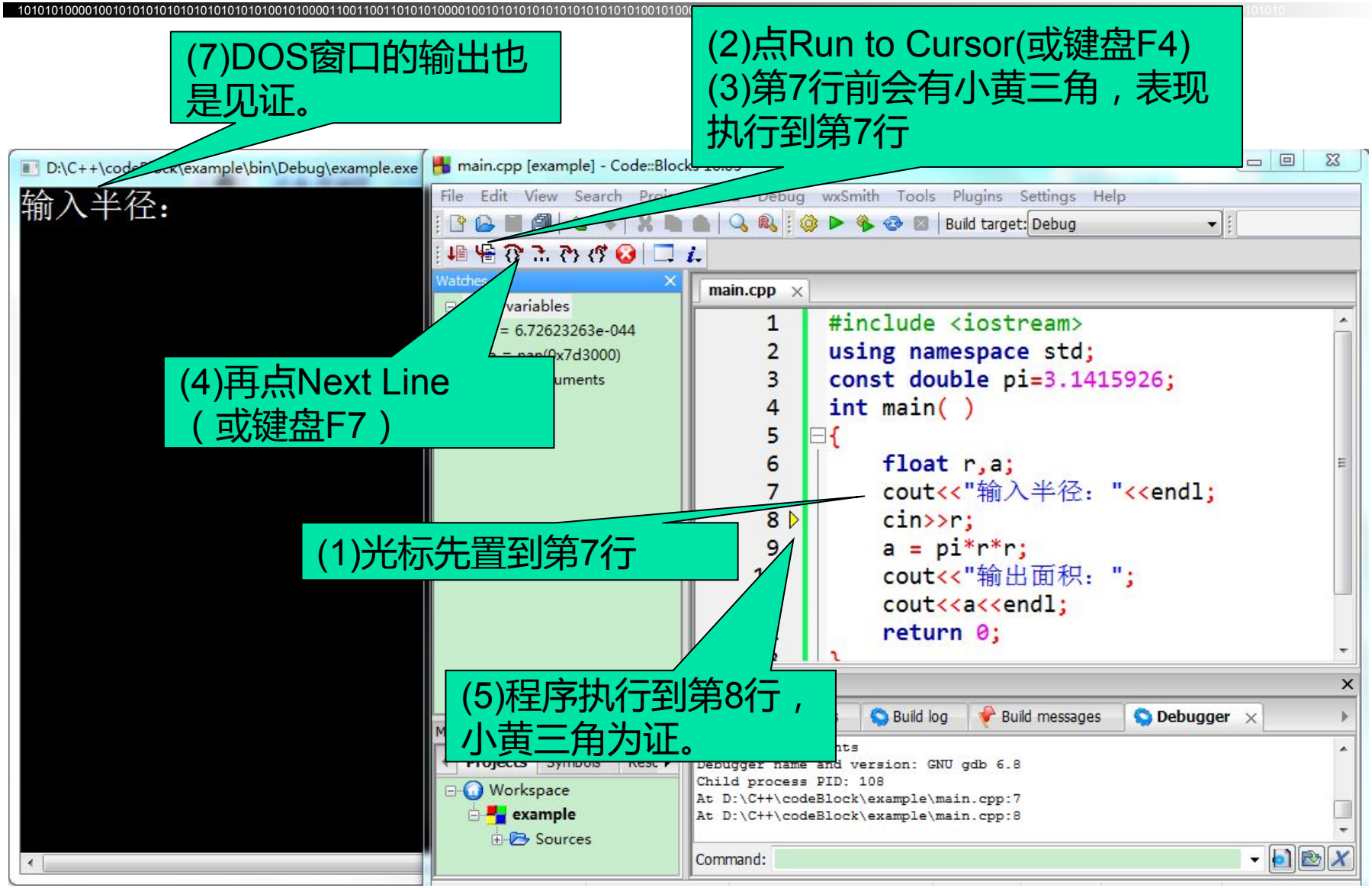
- 记得要边看边练

贺利坚 Email: sxhelijian@163.com

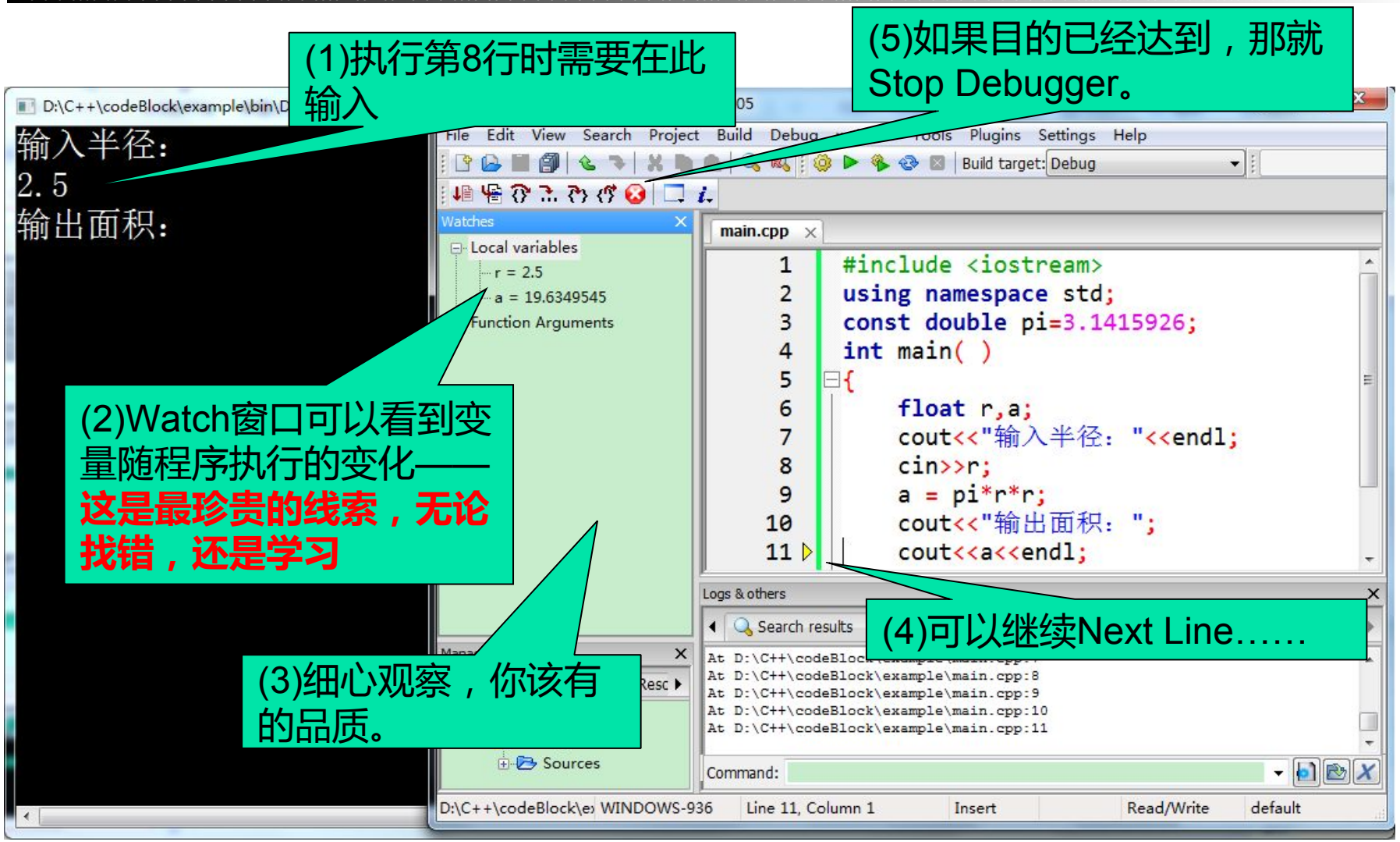
准备


main.cpp [example] - Code::Blocks 10.05
File Edit View Search Project Build Debug wxSmith Tools Plugins Settings Help
Build target: Debug
Management
Projects Symbols Resc
Workspace
example
Sources
main.cpp
1 #include <iostream>
2 using namespace std;
3 const double pi=3.1415926;
4 int main( )
5 {
float r,a;
cout<<"输入半径: "<<endl;
cin>>r;
a = pi*r*r;
cout<<"输出面积: ";
cout<<a<<endl;
return 0;
}
Logs & others
Code::Blocks Search results Build log Build messages
Linking console executable: bin\Debug\example.exe
Output size is 913.62 KB
Process terminated with status 0 (0 minutes, 3 seconds)
0 errors, 0 warnings
D:\C++\codeBlock\e WINDOWS-936 Line 3, Column 27 Insert
(3)最好让调试工具栏出现
(1)设计一个求圆面积的程序
(2)程序经built没有语法错误

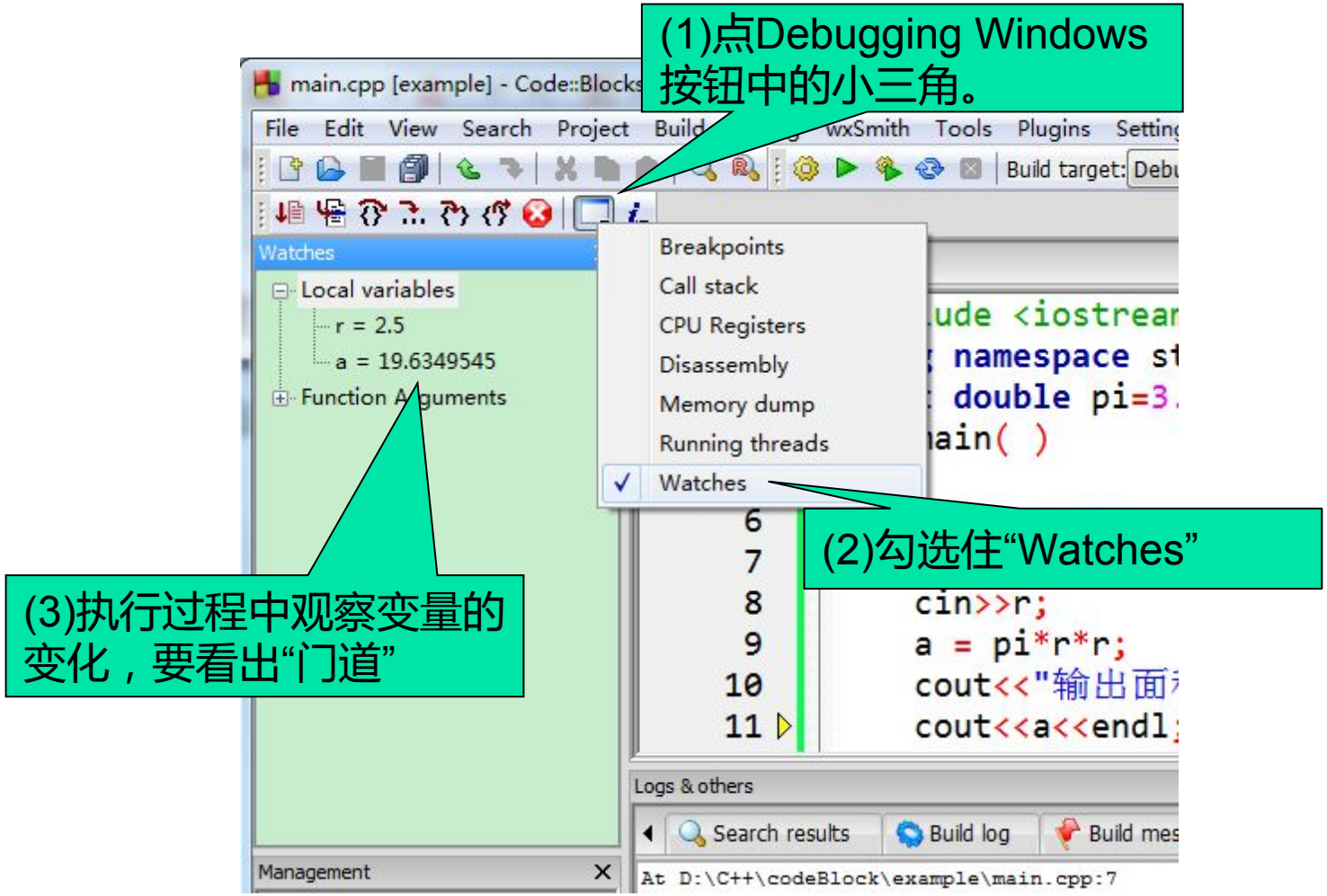
照这样做……


(7)DOS窗口的输出也是见证。
(2)点Run to Cursor(或键盘F4)
(3)第7行前会有小黄三角，表现执行到第7行
(4)再点Next Line（或键盘F7）
(1)光标先置到第7行
(5)程序执行到第8行，小黄三角为证。
输入半径：
main.cpp [example] - Code::Blocks
File Edit View Search Debug wxSmith Tools Plugins Settings Help
Build target: Debug
main.cpp
#include <iostream>
using namespace std;
const double pi=3.1415926;
int main( )
{
float r,a;
cout<<"输入半径： "<<endl;
cin>>r;
a = pi*r*r;
cout<<"输出面积： ";
cout<<a<<endl;
return 0;
}
Build log
Build messages
Debugger
Debugger name and version: GNU gdb 6.8
Child process PID: 108
At D:\C++\codeBlock\example\main.cpp:7
At D:\C++\codeBlock\example\main.cpp:8
Workspace
example
Sources
Command:

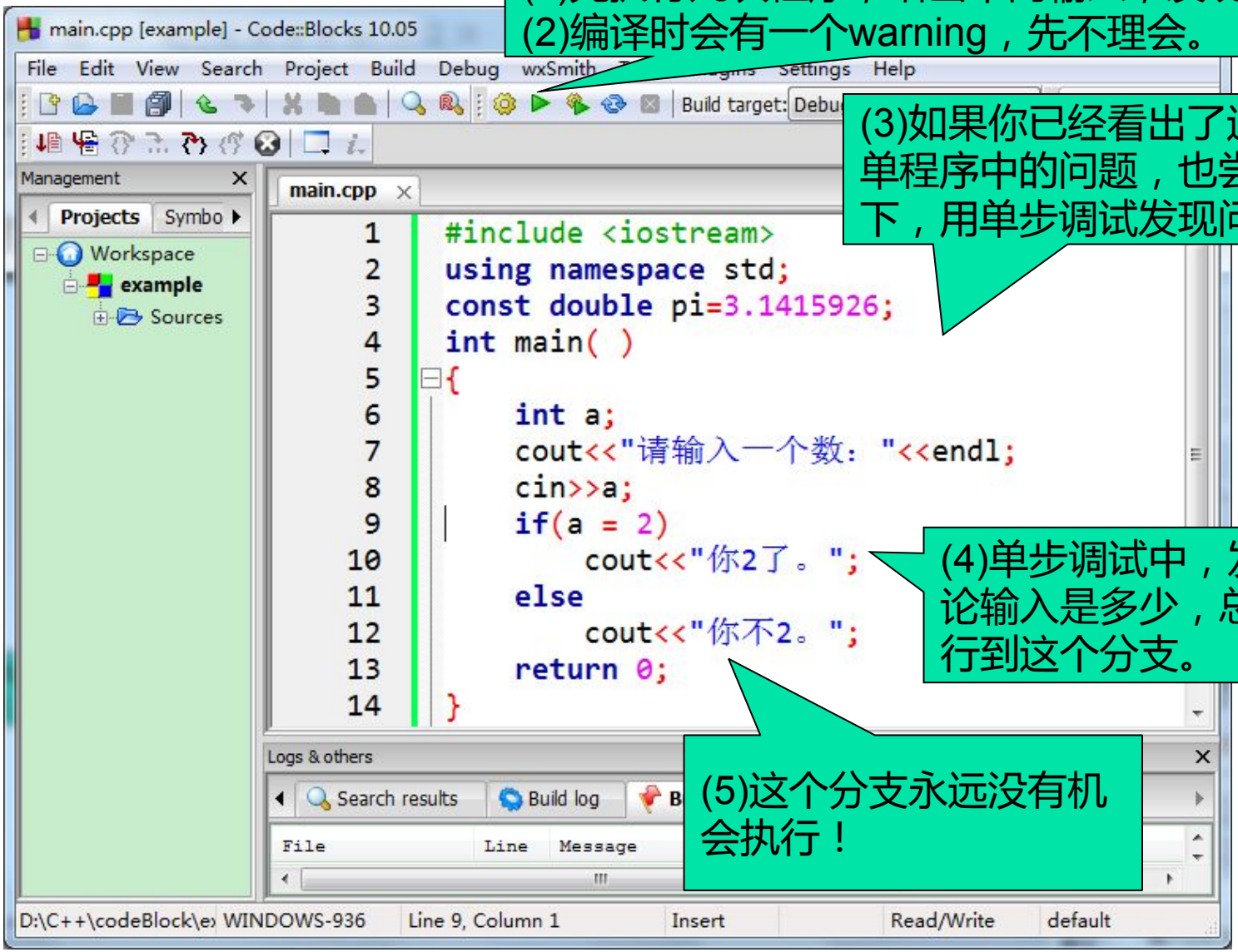
已经Next Line到了第11行



有人找不到Watch窗口……



小试牛刀：实践一下



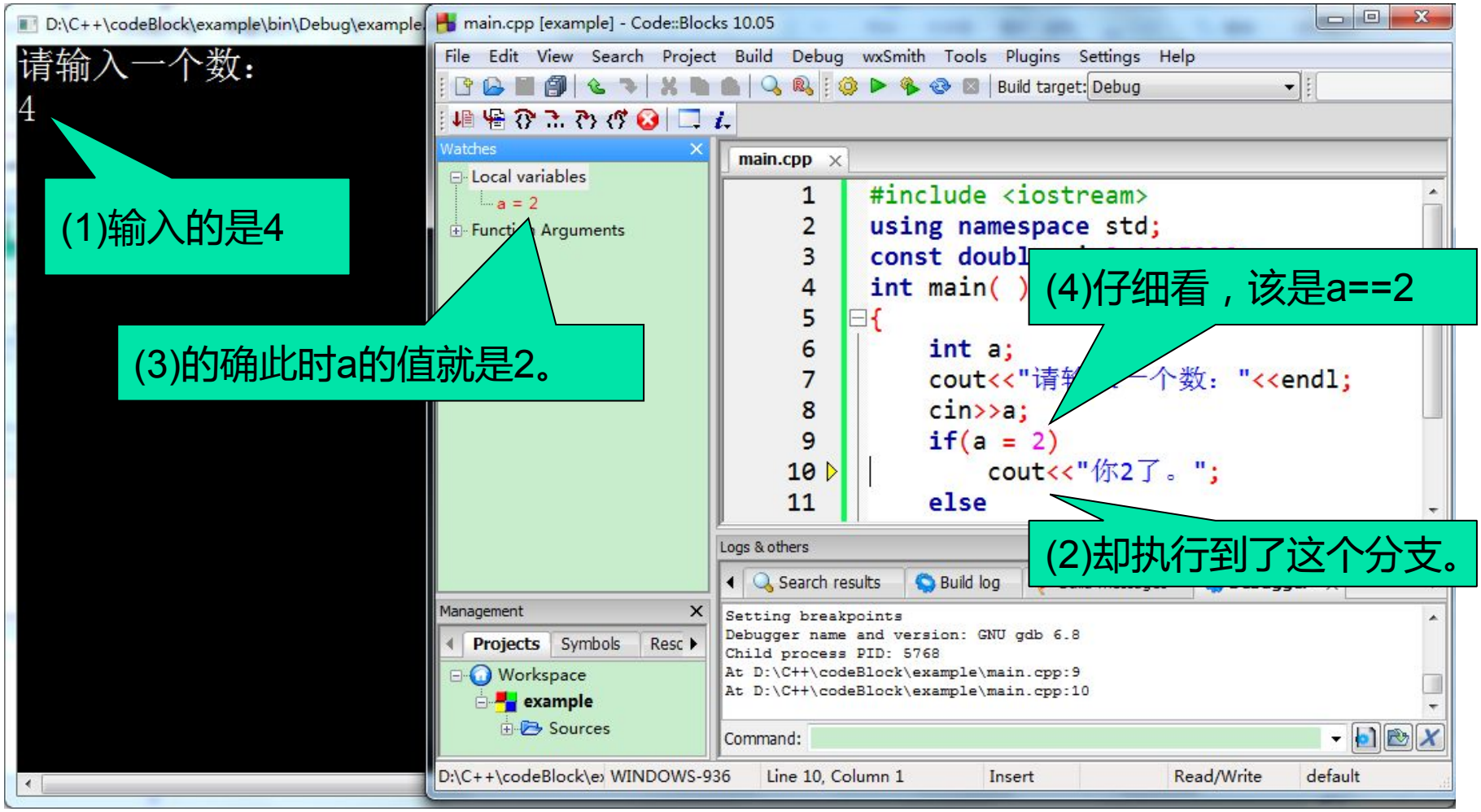
(1)先执行几次程序，给出不同输入，发现总是2。
(2)编译时会有一个warning，先不理会。

(3)如果你已经看出了这个简单程序中的问题，也尝试一下，用单步调试发现问题。

(4)单步调试中，发现无论输入是多少，总会执行到这个分支。

(5)这个分支永远没有机会执行！

真相大白


D:\C++\codeBlock\example\bin\Debug\example
请输入一个数：
4
(1)输入的是4
(3)的确此时a的值就是2。
main.cpp [example] - Code::Blocks 10.05
File Edit View Search Project Build Debug wxSmith Tools Plugins Settings Help
Build target: Debug
Watches
Local variables
a = 2
Arguments
main.cpp
#include <iostream>
using namespace std;
int main( )
{
int a;
cout<<"请输入一个数： "<<endl;
cin>>a;
if(a = 2)
cout<<"你2了。 ";
else
(4)仔细看，该是a==2
(2)却执行到了这个分支。
Logs & others
Search results
Build log
Setting breakpoints
Debugger name and version: GNU gdb 6.8
Child process PID: 5768
At D:\C++\codeBlock\example\main.cpp:9
At D:\C++\codeBlock\example\main.cpp:10
Command:
Management
Projects
Symbols
Workspace
example
Sources
Line 10, Column 1
Insert
Read/Write
default

技能2：跟踪自定义函数

- Step Into和step out出场

贺利坚 Email: sxhelijian@163.com

要跟踪这个程序的执行

```
#include <iostream>
using namespace std;
const double pi=3.1415926;
float area(float r);
int main( )
{
    float r1,a1;
    cin>>r1;
    a1=area(r1);
    cout<<a1<<endl;
    return 0;
}
float area(float r)
{
    float a;
    a = pi*r*r;
    return a;
}
```

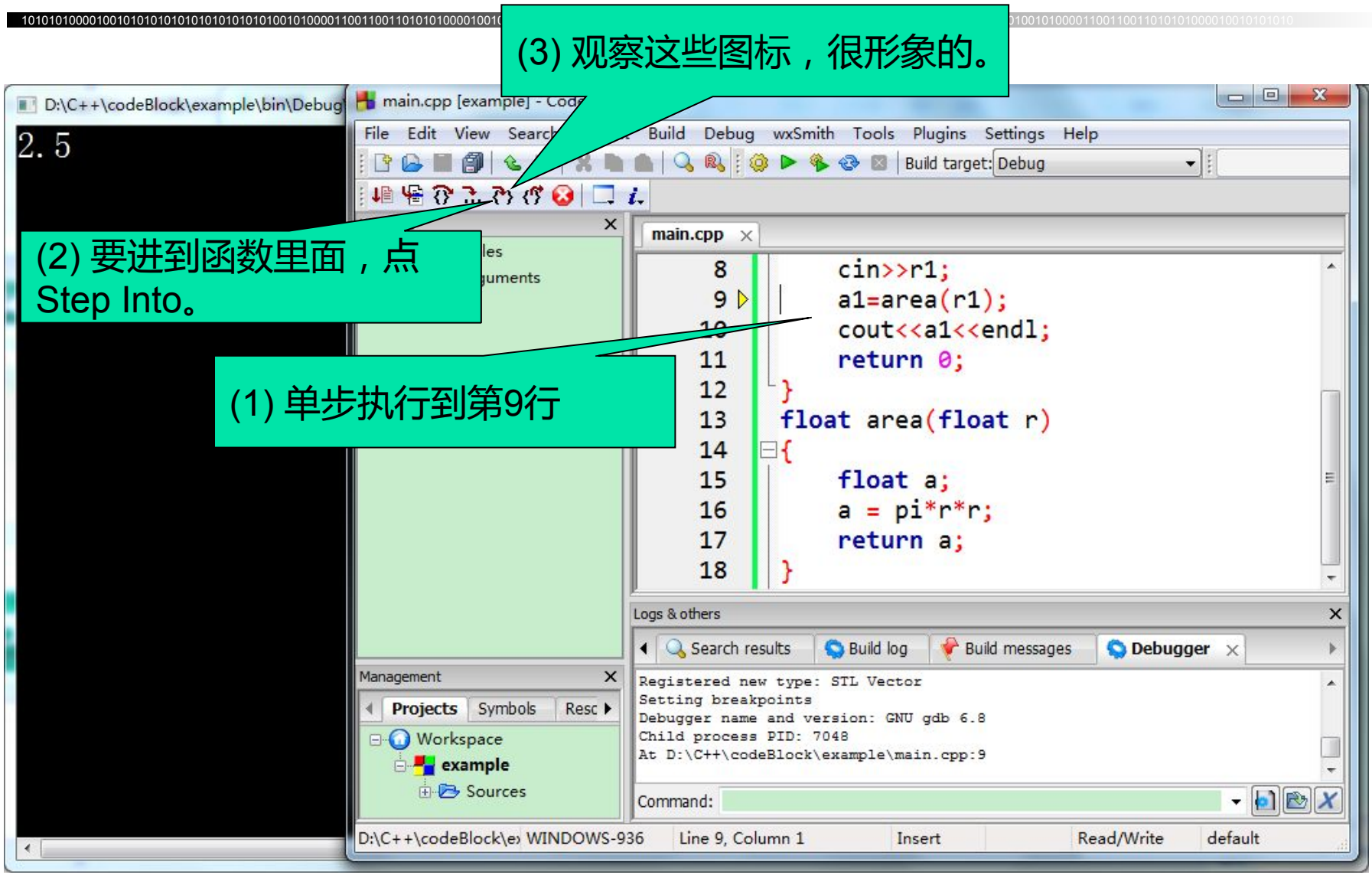
(2) 用前面的办法跟踪时，此行执行Next Line，直接跳到下一行，将函数调用当作一个整体看待

(1) 程序中有了自定义函数

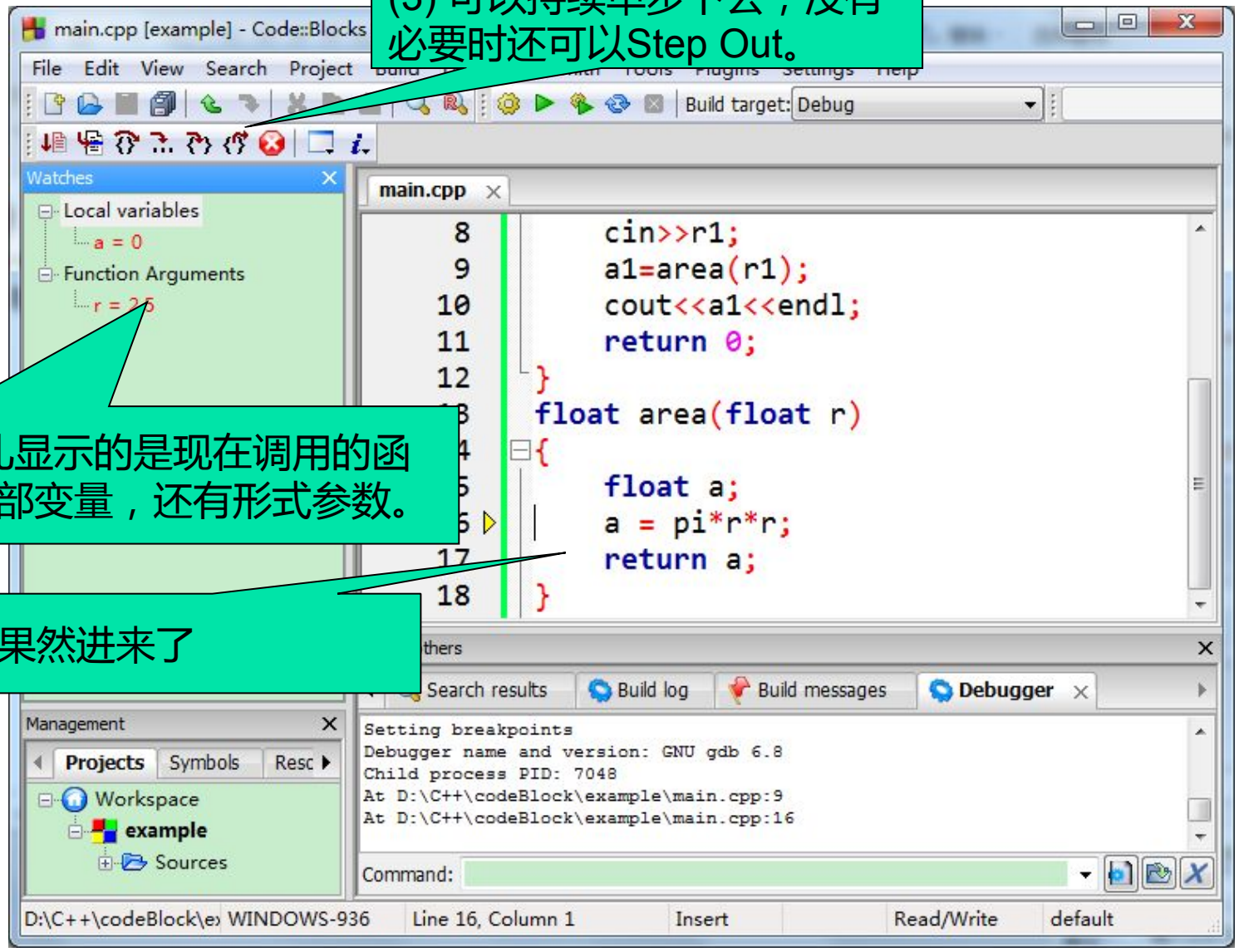
(3) 错误可能发生在自定义函数中，能不能“尾随”“跟踪”进函数内部来？

(4) 期待能够到函数里面看一看。

Step Into到函数中去



现在Into进来了


(3) 可以持续单步下去，没有必要时还可以Step Out。
(2) 这儿显示的是现在调用的函数的局部变量，还有形式参数。
(1) 果然进来了
main.cpp [example] - Code::Blocks
File Edit View Search Project
Build target: Debug
Watches
Local variables
a = 0
Function Arguments
r = 25
main.cpp
8 cin>>r1;
9 a1=area(r1);
10 cout<<a1<<endl;
11 return 0;
12 }
float area(float r)
{
float a;
a = pi*r*r;
return a;
}
Search results
Build log
Build messages
Debugger
Setting breakpoints
Debugger name and version: GNU gdb 6.8
Child process PID: 7048
At D:\C++\codeBlock\example\main.cpp:9
At D:\C++\codeBlock\example\main.cpp:16
Command:
Management
Projects
Symbols
Workspace
example
Sources
Line 16, Column 1
Insert
Read/Write
default

实践，检验技能的唯一标准

```
#include <iostream>
using namespace std;
float max(float x, float y);
int main ()
{
    float a,b,c;
    cin>>a>>b;
    c=max(a, b) ;
    cout<<"The max is "<<c<<endl;
    return 0;
}
float max(float x, float y)
{
    float z;
    z=(x<y)? x : y ;
    return  z;
}
```

- **这个程序本来要输出大值，却总输出小值！**
- 可能问题你已经看出来了，跟踪进函数里面看一看吧。

技能3：会用断点

- 能“一步到位”的办法

贺利坚 Email: sxhelijian@163.com

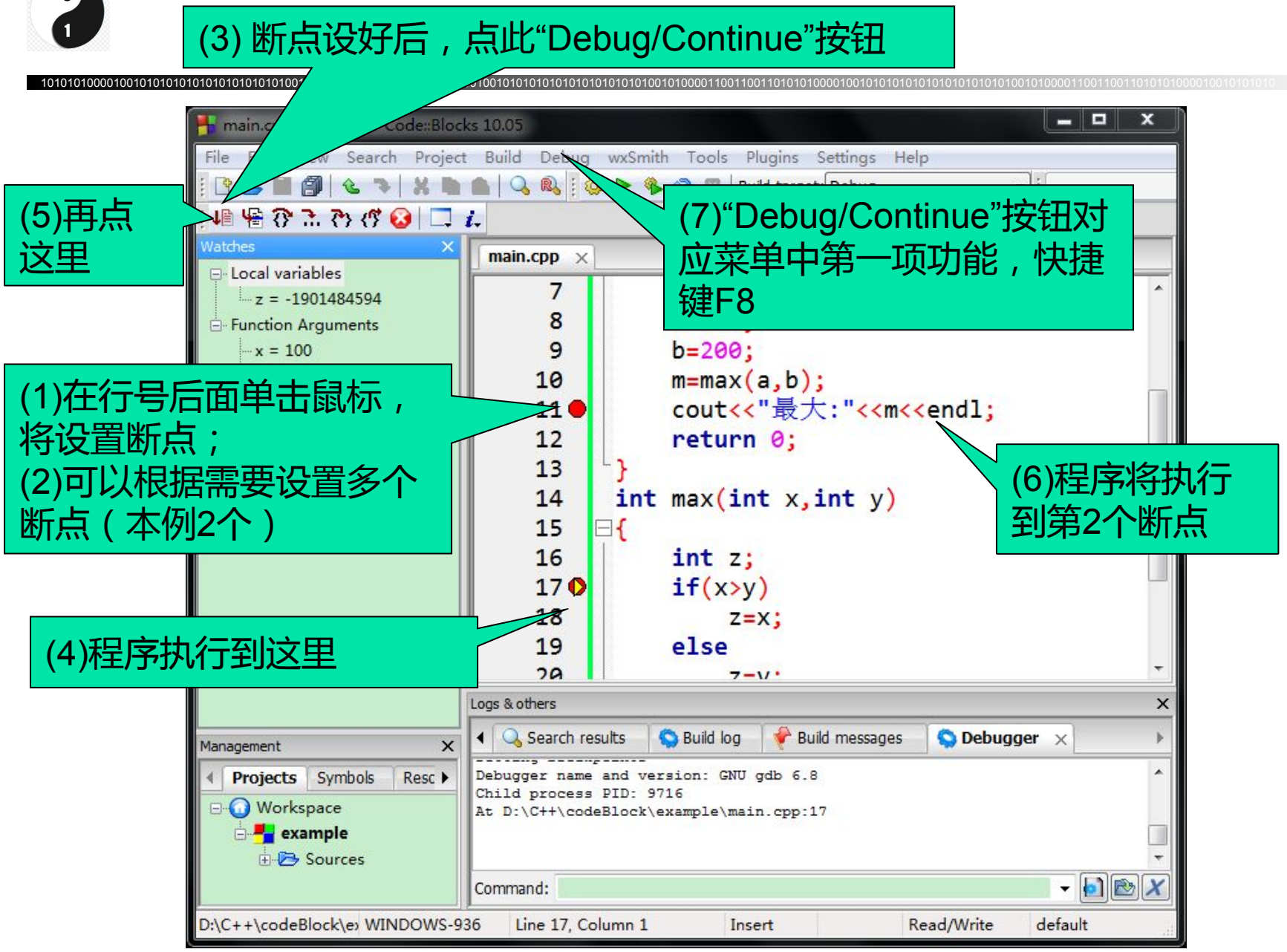
断点(breakpoint)

- 单步执行
  - 每执行一步断一下，看到每一步到底干了些什么
  - 方式：Next Line/Step Into/Step Out等
- 运行到光标
  - 运行到光标所在位置，很方便
  - 光标所在位置，实际就是一个“断点”
  - 方式：Run to Cursor
- 设置断点
  - 任意指定到哪儿断开，想在哪儿停下来，就在哪儿停
  - 相比：单步太烦，运行到光标只能指定一个地方

要跟踪的程序

```
#include<iostream>
#include<cmath>
using namespace std;
int max(int,int);
int main( )
{
    int m,a,b;
    a=100;
    b=200;
    m=max(a,b);
    cout<<"最大:"<<m<<endl;
    return 0;
}
```

```
int max(int x,int y)
{
    int z;
    if(x>y)
        z=x;
    else
        z=y;
    return z;
}
```


(3) 断点设好后，点此“Debug/Continue”按钮
(5)再点这里
(7)“Debug/Continue”按钮对应菜单中第一项功能，快捷键F8
(1)在行号后面单击鼠标，将设置断点；
(2)可以根据需要设置多个断点（本例2个）
(6)程序将执行到第2个断点
(4)程序执行到这里
Code::Blocks 10.05
File Search Project Build Debug wxSmith Tools Plugins Settings Help
Watches
Local variables
z = -1901484594
Function Arguments
x = 100
main.cpp
7
8
9 b=200;
10 m=max(a,b);
11 cout<<"最大:"<<m<<endl;
12 return 0;
13 }
14 int max(int x,int y)
15 {
16 int z;
17 if(x>y)
z=x;
19 else
Logs & others
Search results
Build log
Build messages
Debugger
Debugger name and version: GNU gdb 6.8
Child process PID: 9716
At D:\C++\codeBlock\example\main.cpp:17
Command:
Management
Projects
Symbols
Workspace
example
Sources
D:\C++\codeBlock\e WINDOWS-936
Line 17, Column 1
Insert
Read/Write
default

问题重新发现

```
#include <iostream>
using namespace std;
float max(float x, float y);
int main ()
{
    float a,b,c;
    cin>>a>>b;
    c=max(a, b) ;
    cout<<"The max is "<<c<<endl;
    return 0;
}
float max(float x, float y)
{
    float z;
    z=(x<y)? x : y ;
    return  z;
}
```

- **这个程序本来要输出大值，却总输出小值！**
- 怀疑函数有问题，设置断点，直接进函数内部看一看吧。

结束语

贺利坚 Email: sxhelijian@163.com

- 工欲善其事，必先利其器
- 实践出真知
- 用调试功能，窥得“内幕”，玩程序于股掌之间

- 借此简易讲义入手，
- 其他功能，自主探究，自然会用